NPointFunctions: computation of physical quantities in BSM theories in FlexibleSUSY

Uladzimir Khasianevich

Institut für Kern- und Teilchenphysik, TU Dresden

SUSY 2024

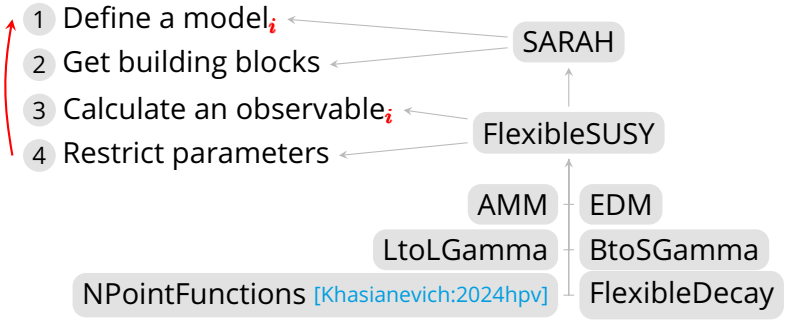
Overview

Structure

1. New essential features in **Part I**
2. Application examples in **Part II**

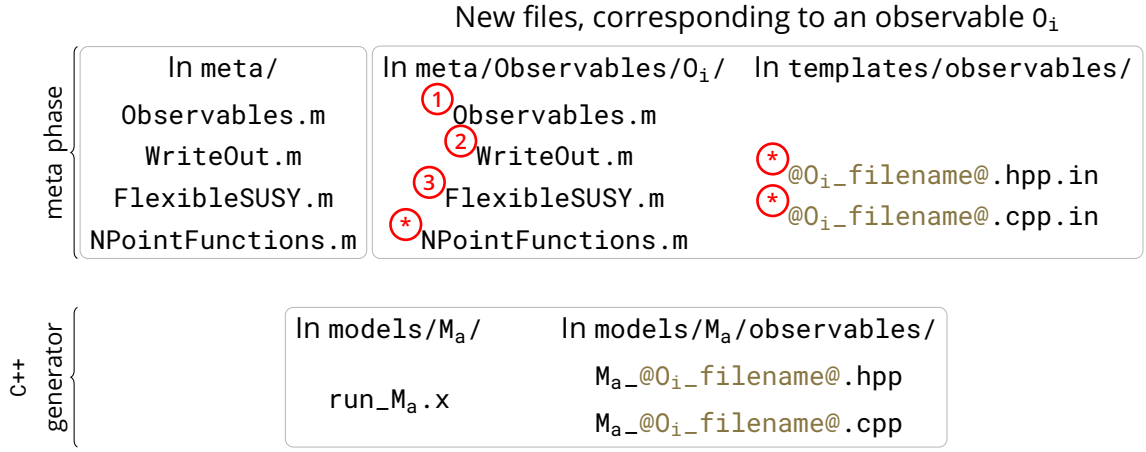
Workflow example (of a phenomenologist)

1. Define a model
2. Get building blocks
3. Calculate an observable
4. Restrict parameters



Part I

Creating new observable



Toy example: output two lepton masses

O_i/Observables.m

```
Observables`DefineObservable[
   FlexibleSUSYObservable`ExampleFermionMass[fermion_[gen_]],
   GetObservableType      -> {2},
   GetObservablePrototype -> "ex_fermion_mass(int gen, auto model, auto qedqcd)"
];
```

O_i/WriteOut.m

```
WriteOut`WriteObservable[
   "FlexibleSUSYLowEnergy", obs:FlexibleSUSYObservable`ExampleFermionMass[_]
] := "Re(observables." <> Observables`GetObservableName[obs] <> "(0))";

WriteOut`WriteObservable[
   "ExampleLeptonMass", obs:FlexibleSUSYObservable`ExampleFermionMass[_]
] := "Re(observables." <> Observables`GetObservableName[obs] <> "(1))";
```

Toy example: output two lepton masses

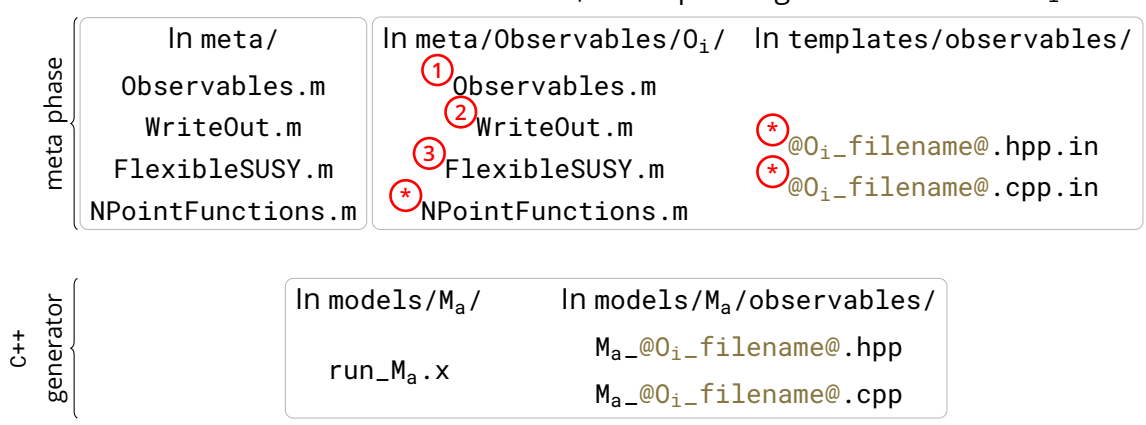
O_i/FlexibleSUSY.m

```
(* Task 1: generate function definition by replacing tokens by concrete C++ code *)
definitions = TextFormatting`ReplaceCXXTokens["
   @type@ @prototype@ {
      return forge<@type@, fields::@fermion@>(gen, model, qedqcd);
   }",
   {"@prototype@" -> Observables`GetObservablePrototype[#], ...}
] &/@ observables;
```

@O_i_filename@.cpp.in

```
template <typename RTYPE, typename FIELD>
auto forge(int idx, ...) {
   ...
   context_mass = context.mass<FIELD>({idx});
   switch (idx) {case 0:  lepton_mass = qedqcd.displayPoleMel(); ...}
   RTYPE res {context_mass, lepton_mass};
   return res;
}
```

Creating new observable



Example to enable NPointFunctions

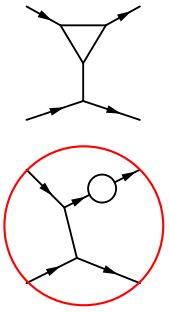
```
(* Task 1: generate function definition by replacing tokens by concrete C++ code *)
npf = NPointFunctions`NPointFunction[
   {field}, (* Incoming particles *)
   {field}, (* Outgoing particles *)
   ...
   NPointFunctions`Observable -> obs[]
];

AppendTo[definitions,
   TextFormatting`ReplaceCXXTokens["
      @type@ @prototype@ {
         const auto npf = npointfunctions::@name@(model, {gen, gen}, {});
         return {npf[0], npf[1]};
      }",
      {"@type@" -> ...}
   ]
];
```

Settings in `NPointFunctions.m`

Settings

```
topologies[LOOPS]
diagrams[LOOPS, TYPE]
amplitudes[LOOPS, TYPE]
order[]
chains[LOOPS]
...
```

Feynman diagrams

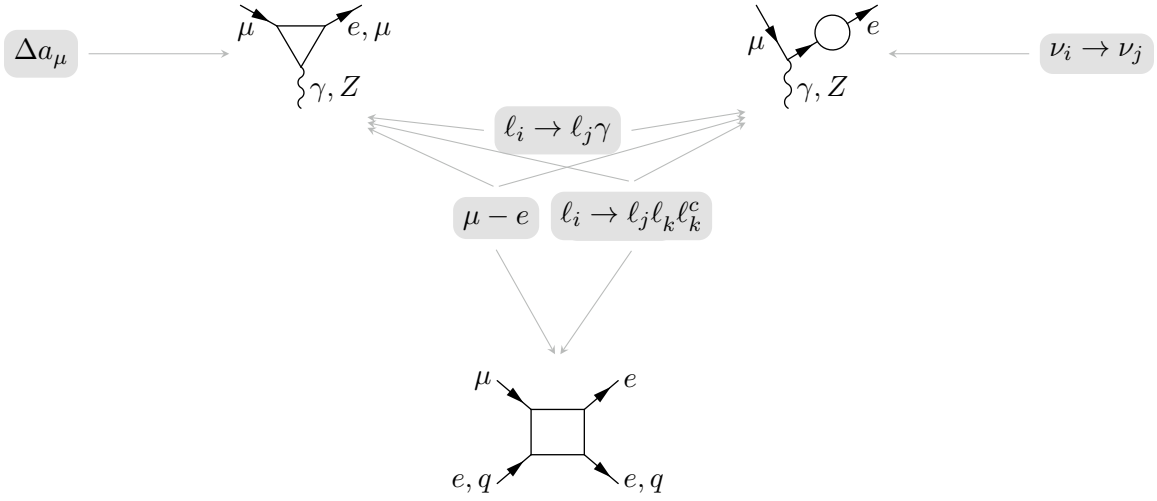


Example

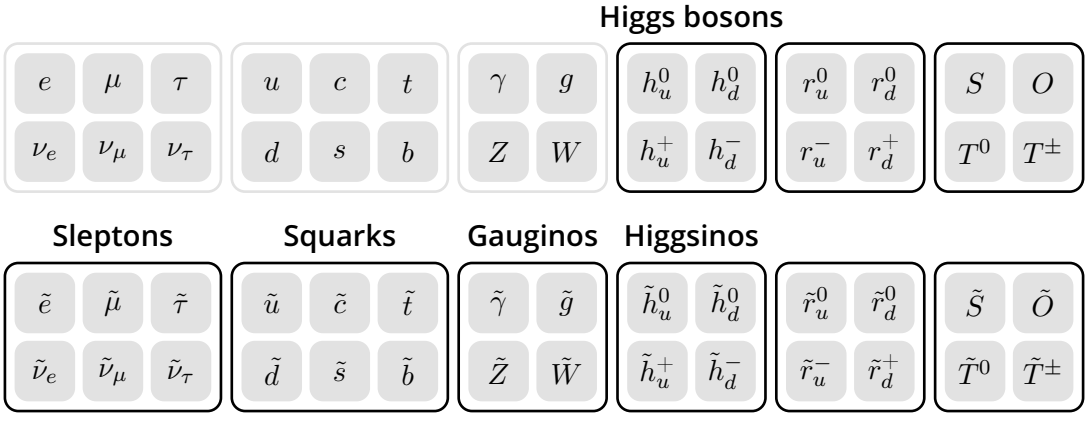
```
topologies[1] = {
   Scalars -> triangleT, Vectors -> outSelfT
};
```

Part II

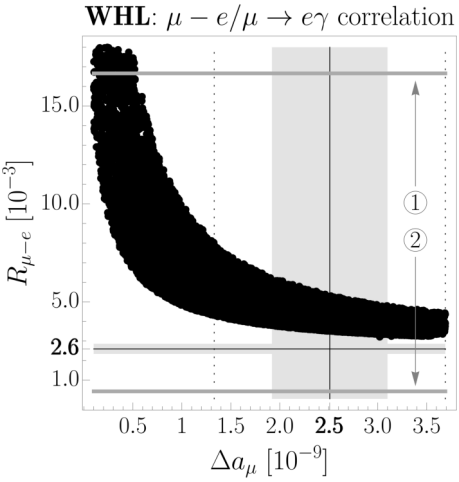
CLFV: Motivation

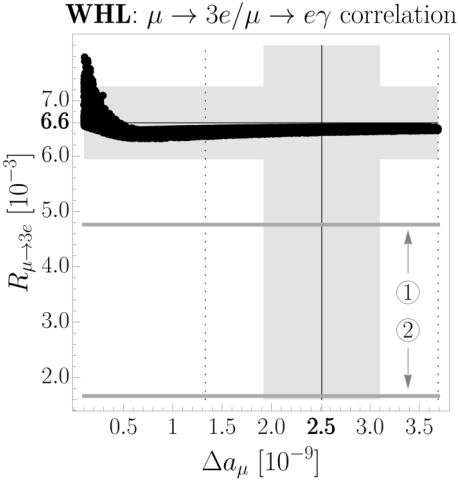


MRSSM: Content

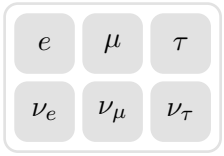


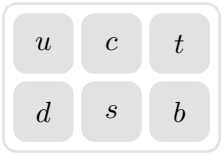
MRSSM: Study example





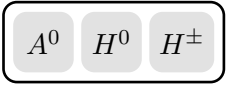
GNM: Content



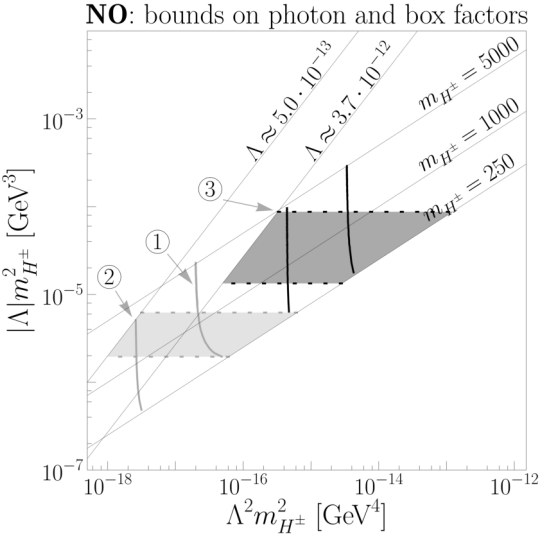


Heavy neutrino

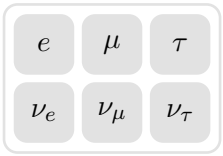
2nd Higgs doublet

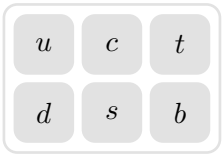


GNM: Study example [Dudenas:2022von, Dudenas:2022xnq]



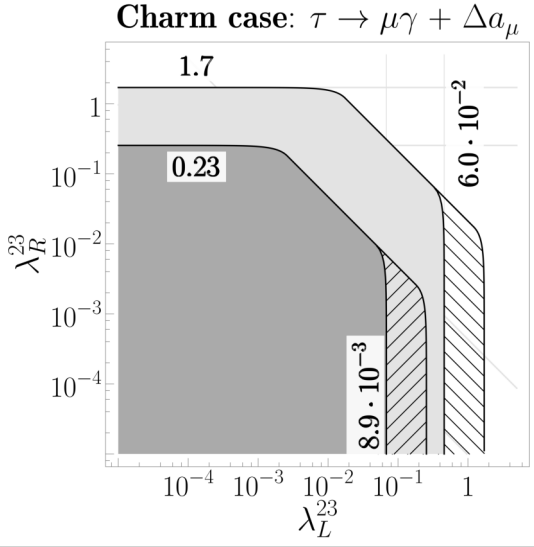
LQ: Content





Scalar leptoquark

LQ: Study example [Khasianevich:2023duu]



Conclusions

- A simpler way to add new observables to FlexibleSUSY
- A way to generate C++ code for Feynman diagrams
- Based on [Khasianevich:2024hpv]

Thank you for attention!